Ministério da
Ciência e Tecnologia

INPE-16672-MAN/54

MANUAL DE USUÁRIO DA PLATAFORMA PLAVIS

Ana Maria Ambrósio
Marcelo Henrique Essado de Morais

Registro do documento original:
<http://urlib.net/sid.inpe.br/mtc-m19@80/2010/02.11.13.49>

INPE
São José dos Campos
2010

PUBLICADO POR:

Instituto Nacional de Pesquisas Espaciais - INPE
Gabinete do Diretor (GB)
Serviço de Informação e Documentação (SID)
Caixa Postal 515 - CEP 12.245-970
São José dos Campos - SP - Brasil
Tel.:(012) 3945-6911/6923
Fax: (012) 3945-6919
E-mail: pubtc@sid.inpe.br

Ministério da
Ciência e Tecnologia

INPE-16672-MAN/54

MANUAL DE USUÁRIO DA PLATAFORMA PLAVIS

Ana Maria Ambrósio
Marcelo Henrique Essado de Morais

Registro do documento original:
<http://urlib.net/sid.inpe.br/mtc-m19@80/2010/02.11.13.49>

INPE
São José dos Campos
2010

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1 - Introdução

A Plataforma para Validação e Integração de Software em Sistemas Espaciais (PLAVIS) caracteriza-se como um produto do projeto de mesmo nome. Esta plataforma agrega, em um único ambiente, protótipos de ferramentas desenvolvidos no âmbito acadêmico pelas instituições participantes do projeto, com o objetivo de disponibilizar ao INPE facilidades para geração automática de casos de testes a partir de especificações baseadas em modelos formais.

A existência de uma plataforma para Validação e Testes de software de sistemas espaciais é de fundamental importância para o INPE. Esta plataforma consolida investimentos na qualidade dos produtos de software através da melhoria da qualidade dos testes, com relação à cobertura da especificação e a eficácia para encontrar falhas, o que depende dos métodos e das técnicas usadas. Essa plataforma não só permitirá estudos de cunho acadêmicos, como a validação de sistemas de software desenvolvidos *in house* e encomendados à indústria, graças à forma sistemática que impõe a atividade de validação.

A primeira versão da plataforma integra as ferramentas: MEGASET, Proteum/FSM, CONDADO e uma implementação do algoritmo UIO [AD1] desenvolvidas pelo Instituto de Ciências Matemáticas e de Computação (ICMC) da USP e pelo Instituto de Computação da Unicamp respectivamente, membros do projeto PLAVIS. Apesar de programarem diferentes algoritmos, todas as ferramentas geram casos de teste a partir de Máquinas de Estados Finitos (MEFs). A capacidade de criar mutantes de especificação em MEFs, executar uma MEF com relação a um conjunto de casos de teste e avaliar conjuntos de casos de teste, também é incorporada a PLAVIS.

A Plataforma PLAVIS é resultado do projeto “PLAVIS - Plataforma para Validação e Integração de Software em Sistemas Espaciais”, financiado pelo CNPq e desenvolvido entre os anos de 2002 e 2004. Este projeto, sob o Número do processo CNPq– 473396/2003-3 foi coordenado pelo Prof. Dr. José Carlos Maldonado e contou com a participação de pesquisadores das universidades: Unicamp, INPE, UfsCar, ICMC/USP e Univem.

Este documento está organizado da seguinte forma: a seção 2 lista os documentos referenciados, a seção 3 traz conceitos relacionados com as técnicas empregadas nas ferramentas de teste da PLAVIS, enquanto que a seção 4 mostra as propriedades da MEF necessárias para que casos de teste sejam gerados, em cada ferramenta da plataforma. Finalmente, as seções 5 e 6 descrevem como usar a plataforma.

2 - Documentos de Referência

[RD1] A. Dahbura and K. K. Sabnani. An experience in estimating fault coverage of a protocol test. In Proc. IEEE INFOCOM'88, pages 71-79, 1988.

[RD2] A. Gill. Introduction to the Theory of Finite-State Machines. McGraw-Hill, New York (1962).

[RD3] Alexandre Petrenko, Nina Yevtushenko. Conformance Tests as Checking Experiments for Partial Nondeterministic FSM. FATES 2005: 118-133.

[RD4] Glenford J. Myers. The At of Software Testing, Second edition. John Wiley & Soons, Inc., 2004.

[RD5] IEEE Std 610.12-1990. Standard Glossary of Software Engineering Terminology.

[RD6] Lucio Fellipe de Mello Neto. Minimização de Conjuntos de Casos de teste para Máquinas de Estados Finitos. Dissertação de Mestrado ICMC-USP março de 2008.

[RD7] Mihalis Yannakakis, David Lee. Testing Finite State Machines: Fault Detection. J. Comput. Syst. Sci. 50(2): 209-227 (1995).

[RD8] Roger S. Pressman. Software Engineering A Practioner´s Approach. Sixth edition, McGraw-Hill, 2005.

[RD9] T. S. Chow. Testing Software Design Modeled by Finite-State Machine. IEEE Transactions on Software Engineering, vol. 4, Issue 3, Pages 178-187. May 1978.

[RD10] E. Martins; S.B. Sabião; A.M. Ambrosio, ConData: a Tool for Automating Specification-based Test Case Generation for Communication Systems, Software Quality Journal, Vol. 8, No.4, 303-319, edited by Anna Liu and Paddy Nixon - Kluwer Academic Publishers, 1999.

[RD11] Weber, T. S. Tolerância a Falhas: conceitos e exemplos. Acessado em 28 de agosto de 2009. Disponível em <http://www.inf.ufrgs.br/~taisy/disciplinas/textos/ConceitosDependabilidade.PDF>.

[R12] SIMOES, A. S. ; AMBROSIO, A. M. ; FABRI, S. C. ; Amaral, A. S. M. S. ; MARTINS, E. ; Maldonado, J.C. . Plavis/FSM: an environment to integrate FSM-based testing tools. In: Simpósio Brasileiro de Engenharia de Software - SBES, 2005, Uberlândia, MG. Anais do Simpósio Brasileiro de Engenharia de Software - Sessão de Ferramentas, 2005. mtc-m16.sid.inpe.br/sid.inpe.br/iris@1916/2005/11.21.18.33

[RD13] Binder, R. Testing Object-Oriented System – Model, Patterns and Tools. Addison-Wesley, 2000.

3 - Conceitos

Esta seção apresenta os conceitos mais importantes aplicados na plataforma PLAVIS.

3.1 - Teste de Software

Teste é um conjunto de atividades que tem como objetivo encontrar falhas, defeitos e erros de um programa por meio de sua execução [RD4]. É uma atividade importante, pois permite que erros sejam identificados e corrigidos antes que o software seja entregue ao usuário final. De acordo com [RD8], a atividade de teste corresponde entre 30 e 40% de todo o esforço empregado no desenvolvimento de software.

As **atividades de teste** consistem basicamente de:

- **Planejamento dos testes** – consiste do levantamento dos objetivos, dos critérios, dos custos, das necessidades e dos prazos para realização dos testes.
- **Projeto dos casos de teste** – atividade que consiste na elaboração de um conjunto de casos de teste que atenda os critérios estabelecidos.
- **Execução dos casos de teste** – consiste da aplicação dos casos de teste, anteriormente criados, contra uma implementação em teste (IUT, do inglês, *Implementation under Test*).

- **Avaliação dos resultados** – esta atividade consiste em comparar se as saídas produzidas durante execução dos casos de teste são conformes as saídas esperadas. Em caso de não conformidade, ações corretivas deverão ser tomadas.

Os termos **defeito, erro e falha**, do inglês, fault, error and failure, tem sido usados de duas formas diferentes pelos grupos de pesquisa: engenharia de software e tolerância a falhas.

Em geral, pesquisadores em Engenharia de Software adotam os seguintes significados, segundo o padrão IEEE [RD5]:

Defeito (fault) - passo, processo ou definição de dados incorretos;

Erro (error) - diferença entre o valor obtido e o valor esperado, ou seja, qualquer estado interno que diverge do estado esperado;

Falha (failure) - produção de uma saída incorreta com relação à especificação. O termo "falha" é utilizado para indicar uma manifestação externa de um erro, ou seja, uma conseqüência.

Pesquisadores em Tolerância a Falhas [RD11] adotam os significados:

Defeito (failure) - é um desvio da especificação. Defeitos não podem ser tolerados, mas deve-se evitar que o sistema apresente defeito.

Erro (error) - é definido que um sistema está em estado errôneo ou em erro se o processamento posterior a partir desse estado pode levar a um defeito.

Falha (fault) - é definido como a causa física ou algorítmica do erro.

Caso de teste consta de elementos do domínio de entrada (D) de um programa P.

Critério de teste é uma diretriz que auxilia o testador a estabelecer um subconjunto T do domínio de entrada D. Um critério de teste é utilizado tanto para avaliar um conjunto de casos de teste quanto para construir os mesmos. Critérios de teste estabelecem requisitos de teste a serem cumpridos. Nota-se a importância da definição dos critérios de teste para realizar uma seleção de casos de teste que aumentem as chances em revelar *defeitos* para estabelecer-se um bom nível de confiança na correção do programa. Exemplos de critérios de teste são: funcional, estrutural, baseada em erros e baseada em estados; exaustivo e aleatório.

3.2 - Máquina de Estados Finitos e suas Propriedades

Uma Máquina de Estados Finitos (MEF) [RD2] é uma máquina hipotética composta por estados e transições. Cada transição liga um estado **a** a um estado **b** (**a** e **b** podem ser o mesmo estado). A cada instante, uma máquina pode estar em apenas um de seus estados. Em resposta a um evento de entrada, a máquina gera uma saída e muda de estado. Tanto a saída gerada quanto o novo estado são definidos unicamente em função do estado atual e do evento de entrada. Uma MEF pode ser representada tanto por um diagrama de transição de estados quanto por uma Tabela de Transição. Em um diagrama de transição de estados, os estados são representados por círculos e as transições são representadas por arcos direcionados entre estados.

Na Figura 1 tem-se a representação de uma MEF para um extrator de comentários de programas em linguagem C [RD9]. A MEF possui 4 estados. Cada arco é rotulado com a entrada que gera a transição e a saída que é produzida, usando o formato "**entrada : saída**".

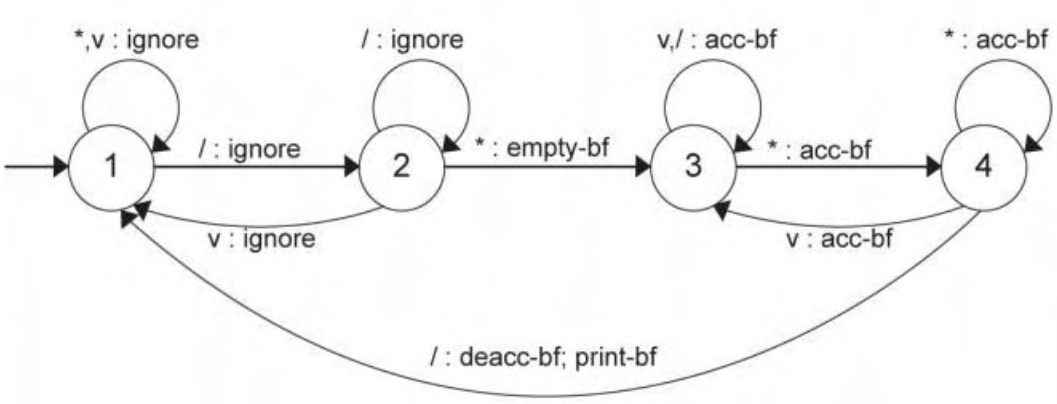

Figura 1 - MEF para um extrator de comentários.

Formalmente, uma MEF M é representada por uma 7-tupla (**S, s_0, X, Y, D_M, δ, λ**), onde:

S - é um conjunto finito de estados, incluindo o estado inicial **s_0**;

X - é um conjunto finito de entradas;

Y - é um conjunto finito de saídas;

D_M - está contido em **S × X** - é o domínio da especificação;

δ - é uma função de transição de estados: ***δ: S x X → S***

λ - é uma função de saída: **λ : D_M → Y**

3.2.1 - Propriedades de uma MEF

As propriedades de uma máquina de estados finitos são:

Completamente especificada (ou completa) – uma máquina completa possui transições definidas para todos os símbolos de entrada em cada um dos estados da MEF. Formalmente, uma MEF é completa se $D_M = S \times X$.

Parcialmente especificada (ou parcial) – quando a máquina não é completa.

Inicialmente conexa - quando todos os estados são alcançáveis a partir do estado inicial s_0.

Fortemente conexa - se para cada par de estados s_i , s_j Є S existe uma sequência de entradas que leva a MEF M do estado s_i ao estado s_j.

Determinística - se em cada estado, dada uma entrada, há somente uma transição definida para um próximo estado. **Não-determinística** - quando a máquina não é determinística.

Mínima - se o número de estados em M é menor ou igual ao número de estados para qualquer máquina M', que seja equivalente a M. Mínima pode ser considerada também reduzida.

M é equivalente a M' - se para cada sequência de entradas aplicada a M´ e a M, M e M´ respondem com uma saída idêntica.

3.3 - Teste baseado em Estados

No teste baseado em estados, uma representação baseada em máquina de estados finitos é utilizada para modelar o aspecto comportamental do sistema ou unidade que será testada. Este tipo de teste consiste na geração de um conjunto de casos de teste a partir do modelo visando encontrar defeitos na implementação correspondente.

Com isso, é possível verificar se a implementação da MEF está ou não de acordo com sua especificação. O teste baseado em estados considera que tanto a especificação quanto a implementação sejam modeladas por uma MEF. Assim, a implementação contém um defeito no caso em que possuir um comportamento diferente em relação à especificação. Os defeitos podem ser de:

- **Inicialização**: quando o estado inicial não é o correto.
- **Transferência** (*transfer faults*): quando o estado atingido por uma transição não é o correto.
- **Saída** (*output faults*): quando a saída gerada por uma transição não é a correta.
- **Estados:**
  a) **Faltantes** (*missing states*): quando os estados da implementação precisam ser aumentados para torná-la equivalente à especificação.
  b) **Extras** (*extra states*): quando os estados da implementação precisam ser reduzidos para torná-la equivalente à especificação.

Cabe observar que um **caso de teste** gerado a partir de uma MEF M consta de uma seqüência de entradas (um subconjunto de X) de M.

3.4 - Métodos de geração de casos de teste

Diversos métodos de derivação de casos de teste a partir de MEFs têm sido propostos. Esses métodos objetivam verificar se uma programação está correta em relação à sua especificação.

Embora os métodos possuam um objetivo comum (o de verificar se uma programação está correta com relação a sua especificação), eles diferem com relação ao custo da geração das sequências de teste, ao tamanho do conjunto de teste, a capacidade de detecção de falhas (efetividade) e as propriedades requeridas para as MEFs.

Figura 2 ilustra a MEF usada na descrição dos métodos abaixo. Essa MEF possui quatro estados {S1, S2, S3, S4}, sendo S1 o estado inicial, as entradas X = {x, y} e as saídas Y = {0, 1}.

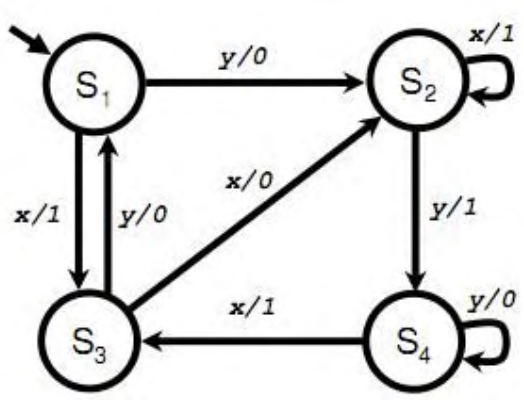

Figura 2 - Exemplo de MEF extraído de Dorofeeva et al. (2005a).

3.4.1 - Método UIO

O método UIO (Unique Input/Output) [RD1] consiste em duas fases.

1a Fase: Verifica cada estado da MEF com sua respectiva *seqüência UIO*. Essa fase consiste na concatenação do *conjunto Q* com a respectiva seqüência UIO de cada estado atingido.

Uma **seqüência UIO** de um estado s_j é uma seqüência de entrada/saída única para esse estado. Em outras palavras, com a aplicação da seqüência UIO pode-se distinguir o estado s_j de qualquer outro estado, pois a saída produzida é específica (única) do estado s_j. Desse modo, a seqüência UIO pode determinar em qual estado a máquina se encontrava antes de sua aplicação.

Para a MEF da Figura 2 consideram-se as seqüências UIO {yy, y, x, yyy} para os estados S1, S2, S3 e S4, respectivamente. Para simplificar, a notação $UIO(S_i)$ representa a seqüência UIO do estado S_i.

Um **conjunto state-cover Q** de uma MEF M com *n* estados é definido como um conjunto com *n* seqüências de transferência, incluindo a seqüência vazia ε, que leva M a partir de seu estado inicial s_0 para cada um dos estados.

Considera-se o conjunto *state-cover* Q = {α_1, α_2, α_3, α_4} tal que α_1, = ε, α_2 =y, α_3 = x, α_4 = yy, para o exemplo da figura 2.

Nesta fase as seqüências geradas são: $\alpha_1 UIO(1)$, $\alpha_2 UIO(2)$, $\alpha_3 UIO(3)$, $\alpha_4 UIO(4)$. Realizando as devidas substituições, essa fase resulta nas seqüências {yy, yy, xx, yyyyy}.

2a Fase: Verifica se cada transição da MEF produz a saída desejada. Nessas transições, cada estado atingido também é verificado pela aplicação de sua respectiva seqüência UIO.

Nesta fase as seqüências são geradas pela concatenação: $\alpha_{i.z} UIO(S_j)$ para todo z Є X sendo que α_i leva a MEF do estado inicial ao estado S_i. Essa fase resulta nas seqüências {xx, yy, yxy, yyyyy, xxy, xyyy, yyxx, yyyyyy}.

Com a retirada das seqüências que são prefixos de outras, o método UIO gera o conjunto de teste TSUIO = {ryxy, rxxy, rxyyy, ryyxx, ryyyyyy} de tamanho 25.

Este método não garante uma cobertura completa de falhas.

3.4.2 - Método W

Um dos métodos mais conhecidos para a geração de seqüências de teste é o Método W (*Automata Theoretic*) proposto por Chow [RD9]. O método W não é aplicado a MEFs parciais. O método requer as seguintes propriedades para a máquina: fortemente conexa, completa, mínima e determinística. Esse método consiste em gerar dois conjuntos de seqüências e concatená-las. Esses dois conjuntos são:

P: conjunto de seqüências que percorre cada transição ao menos uma vez.

T: conjunto de seqüências capaz de identificar qual é o estado da máquina.

O conjunto T é gerado a partir de um *conjunto de caracterização*.

Um **conjunto de caracterização** (*characterization set*), freqüentemente chamado de conjunto W, é um conjunto de seqüências de entrada que possui uma seqüência que diferencia todo par de estados existentes em M. O conjunto de caracterização sempre existe para MEFs reduzidas.

Em seguida, é estimado o número *m* de estados da máquina a ser testada. Se o número estimado for igual ao número de estados *n* da máquina real, então T = W, senão tem-se $T = \bigcup_{i=0}^{m-n}(X^i W)$, onde $X^0 = \{\in\}$ e $X^i = XX^{i-1}$ .Ao fim, a seqüência de teste gerada dá-se pela concatenação de P com T. As seqüências desse conjunto são executadas uma a uma na máquina, gerando as saídas que são analisadas posteriormente.

Em suma, o método W consiste em três passos principais:

1. Estima-se um número máximo (m) de estados que a implementação possa conter.

2. Geram-se as seqüências de teste que garantem que cada transição foi implementada corretamente.

3. Verificam-se as respostas geradas pelas seqüências de teste produzidas na segunda etapa.

Se a implementação da MEF (a IUT, *Implementation under Test*) gerar saídas corretas a partir das seqüências de entrada geradas pelo método W, esta máquina está correta, pois o método é confiável para testar estruturas de controle modeladas por uma MEF [RD9].

Contudo, o método W produz muitas seqüências de entrada para serem testadas, o que pode promover um alto custo para a realização da etapa de teste.

A aplicação do método W na MEF da Figura 2, considerando m = n tem-se:

- o conjunto T = W = {x, y, yy}.
- o conjunto **transition cover** P = {ε, x, y, xx, xy, yy, yx, yyy, yyx}.
- pela concatenação de P com T obtém-se as seqüências:

{x, y, yy, xx, xy, xyy, yx, yy, yyy, xxx, xxy, xxyy, xyx, xyy, xyyy, yyx, yyy, yyyy, yxx, yxy, yxyy, yyyx, yyyy, yyyyy, yyxx, yyxy, yyxyy}.

Com a retirada das seqüências que são prefixos de outras, a aplicação do método W na MEF da Figura 3.4 resulta no conjunto TSW = {rxxx, rxxyy, rxyx, rxyyy, ryxx, ryxyy, ryyxx, ryyxyy, ryyyx, ryyyyy} de tamanho 49.

3.4.3 - Método State Counting

O método *State Counting* (SC), [RD3], atinge os mesmos objetivos do método W em relação à efetividade, ou seja, o método garante a cobertura completa de falhas existentes na implementação e ainda aceita uma MEF parcial.

De modo geral, o método *State Counting* utiliza um algoritmo que expande as seqüências de teste a partir de um estado da MEF até que seja atingida uma condição que permita verificar que todas as falhas já foram identificadas. Por exemplo, se um estado é visitado mais do que *m* vezes, sendo ***m*** o número de estados da MEF, pode-se parar de expandir a seqüência, uma vez que, desse ponto em diante, o comportamento começará a se repetir.

O método *State Counting* gera um conjunto de casos de teste completo a partir de MEFs parciais e não reduzidas.

Considerando a MEF da Figura 2, o método *State Counting* gera o conjunto TSSC = {rxxx, rxxy, rxyx, rxyyy, ryxx, ryxy, ryyxx, ryyyx, ryyyyy} de tamanho 41.

3.4.4 - Método Switch-cover

O método *switch-cover* foi proposto por Chow em 1978 [RD9] e é descrito por Binder [RD13] como "visita as transições (*transition tours)* de M produzindo uma sequência de entradas partindo-se de S_0, seu estado inicial, e continua visitando cada transição pelo menos uma vez, para depois retornar para S_0 ou para um estado final de M". Uma visita ou *tour* (chamada por Binder 2000 de *round-trip-path*) é uma sequência de transições. O método consiste de: (a) executar uma busca em profundidade (*depth-first serach traversal*) para obter os caminhos existentes, considerando-se apenas uma interação de um *loop*, se o *loop* existir e (b) produzir uma árvore de transição (*transition tree*).

A árvore de transição inicia no estado inicial (S_0) como o nó raiz (*root node*). Do estado inicial, uma aresta é desenhada para cada transição e um nó é adicionado para cada próximo estado. Um nó folha, isto é, um estado sem arestas, corresponde a um estado que já foi adicionada a árvore ou a um estado final. Este procedimento é repetido até que todas as transições tenham sido visitadas. A estrutura da árvore depende da ordem na qual o algoritmo percorre as transições.

O método requer que (i) um estado inicial seja indicado pelo usuário, pois nem sempre é possível detectá-lo automaticamente [RD7] e (ii) M seja uma máquina de Mealy e conectada.

O algoritmo *transition tour*, implementado na ferramenta Condado [RD10], suporta o método switch-cover, sendo, portanto, mais exaustivo do que os métodos *transitons tours* tradicionalmente utilizados, pois, ele cobre todos os *caminhos* alcançáveis a partir do estado inicial e não apenas visita todas as transições da MEF.

3.5 - Mutantes

O Teste de Mutação é um critério que se baseia nos erros que podem ser cometidos ao longo do processo de desenvolvimento do software. Para modelar esses erros, um conjunto de operadores de mutação é definido e aplicado ao produto em teste (P), gerando versões modificadas do produto (m), chamadas de mutantes.

A aplicação deste critério envolve as seguintes atividades: (i) execução do programa com um conjunto de casos de teste T, (ii) geração dos mutantes de P, (iii) execução dos mutantes com T e (iv) análise dos resultados da execução de T contra os mutantes.

Os mutantes são gerados a partir de pequenas perturbações aplicadas em P. Tanto P quanto os mutantes são testados com um conjunto de casos de teste T. Caso o comportamento de m (um mutante de P) seja diferente de P, então esse mutante é dito "morto". Caso contrário, esse mutante está "vivo".

Análise de Mutantes – tem por objetivo determinar um conjunto de casos de teste que consiga revelar, através da execução de P, as diferenças de comportamento existentes entre P e seus Mutantes.

Operadores de mutação – definem as regras para alterar P, caracterizando um conjunto de implementações (ou de modelos) alternativas. Definem-se operadores de mutação para MEFs, a qual sofre a alteração quando o operador é aplicado. Os operadores de mutação de MEFs usados na PLAVIS são descritos na seção 5.3.

Mutante equivalente - m é dito equivalente a P, se para qualquer caso de teste de T o comportamento dos dois programas não difere.

Escore de mutação – é um valor no intervalo [0,1] que fornece a medida de quanto o conjunto de casos de testes analisado aproxima-se da adequação. Quanto mais próximo esse valor chegar de 1, melhor é o seu conjunto de casos de testes. O escore de mutação é computado como:

$$Sm\ (\mathrm{P},\mathrm{T}) = \frac{muttes\_mortos}{muttes\_criados - muttes\_equivalentes}$$

Muttes_criados = número de mutantes gerados a partir de P.

Muttes_mortos = número de mutantes, gerados a partir de P, que foram mortos com o conjunto de casos de teste T.

Muttes_equivalentes = número de mutantes, gerados a partir de P, considerados equivalentes.

4 - Propriedades das máquinas para aplicação dos métodos

Esta seção relaciona os métodos e as propriedades requeridas da máquina para que o método seja aplicado, ver Tabela 1.

Tabela 1 - Métodos e propriedades das máquinas de estado.

Método	Função	Propriedades requeridas da MEF
Análise de Mutantes	geração dos mutantes, execução de P com um conjunto de casos de teste T, execução dos mutantes com T, análise dos mutantes	-

Switch-cover set	geração de casos de teste	Fortemente conexa
UIO	geração de casos de teste, provendo casos de teste com combinação de entradas.	Inicialmente conexa, determinística, e mínima.
W	geração de casos de teste, provendo casos de teste de conformidade estrutural da MEF	Inicialmente conexa, completamente especificada, mínima e determinística.
State-counting	garante a cobertura completa de falhas existentes na implementação de uma MEF parcial.	Inicialmente conexa e determinística.

5 - Uso da Plataforma PLAVIS

Esta seção descreve como usar a plataforma PLAVIS [RD12].

5.1 - Acesso a PLAVIS

A PLAVIS é acessada remotamente através do navegador de internet como, por exemplo, o **Internet Explorer**, no endereço:

http://www2.dem.inpe.br/plavisFSM/

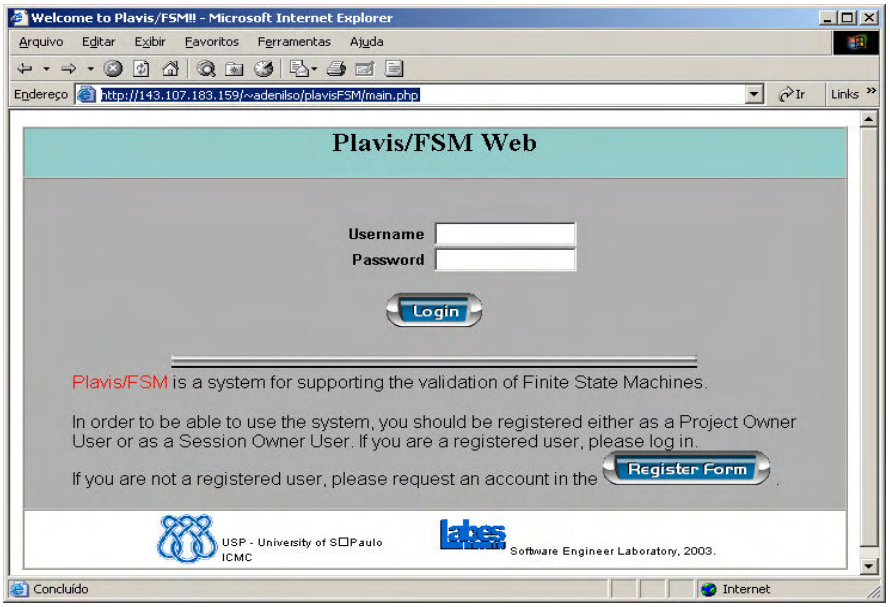

Figura 3 - Página de Login

Para obter acesso a PLAVIS, faz se necessário estar registrado, para tanto use o botão *Register Form*, assim que for confirmado o seu registro, um e-mail com o *Username* e *Password* será enviado para o e-mail fornecido.

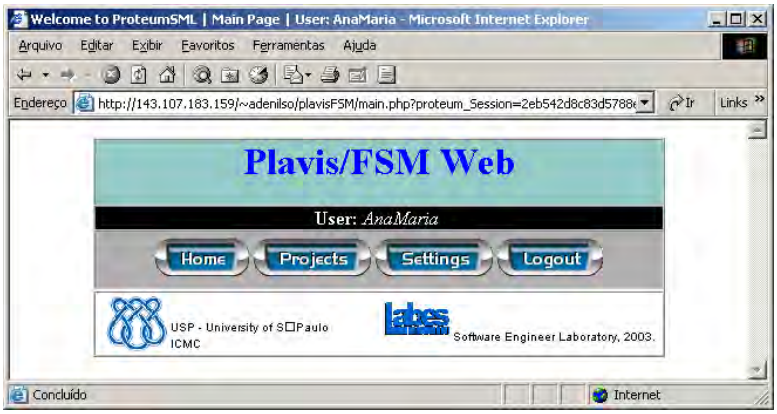

Figura 4 - Tela Inicial

Botões:

Home*:* este botão é apresentado em todas as telas da plataforma PLAVIS, podendo ser acessado a qualquer momento para retornar a *Tela Inicial* da plataforma.

Projects*:* este botão exibe os projetos em desenvolvimento e os finalizados, também são exibidos os projetos criados por outros usuários da plataforma. Entretanto o acesso ao projeto é restrito ao usuário criador.

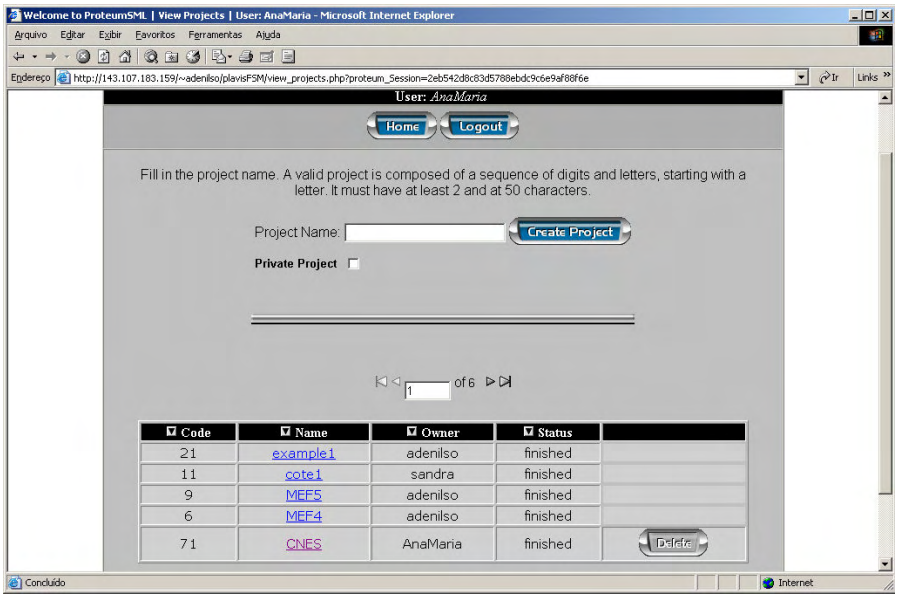

Figura 5 - Tela Projects

Settings*:* este botão apresenta a *Tela Settings* com recursos para modificar a visualização das telas e a quantidade de mutantes que serão exibidos a cada tela. As modificações feitas aqui terão efeito em todos os projetos em desenvolvimento ou que se encontram prontos.

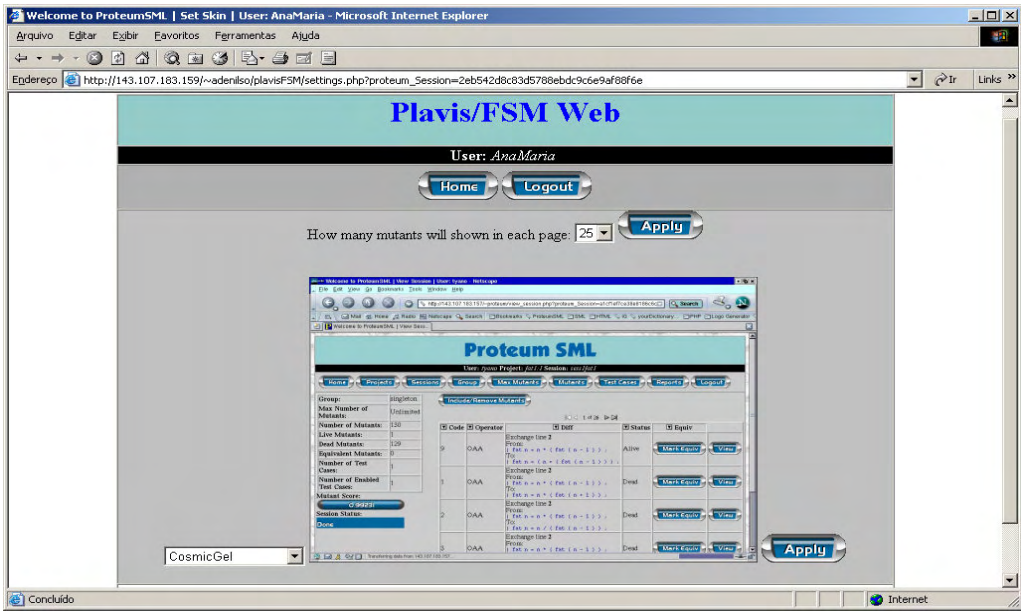

Figura 6 - Tela Settings

Logout*:* este botão é exibido em todas as telas da plataforma PLAVIS, possibilitando o uso a qualquer momento para sair da plataforma PLAVIS.

5.2 - Criação de um projeto

Como citado, dentro da Tela *Projects* podemos visualizar os projetos já existentes ou criar novos projetos.

Para criar um novo projeto, basta escrever na caixa de texto o nome, como por exemplo, “NovoProjeto” e clicar no botão *Create Project*, o nome do projeto deve estar na lista abaixo da caixa de texto. Observe que o *Status* é igual à *initial*, significa que o projeto esta sendo editado.

Obs.: para os nomes atribuídos aos projetos, não podem conter caracteres especiais, tais como: ! # ( ) e o primeiro caractere não pode ser numérico. Porém permite o uso de caracteres em maiúsculos e minúsculos e números no meio ou final do nome.

Para acessar o projeto criado basta clicar em sobre o nome dado ao projeto durante a criação, em seguida exibirá a tela do projeto:

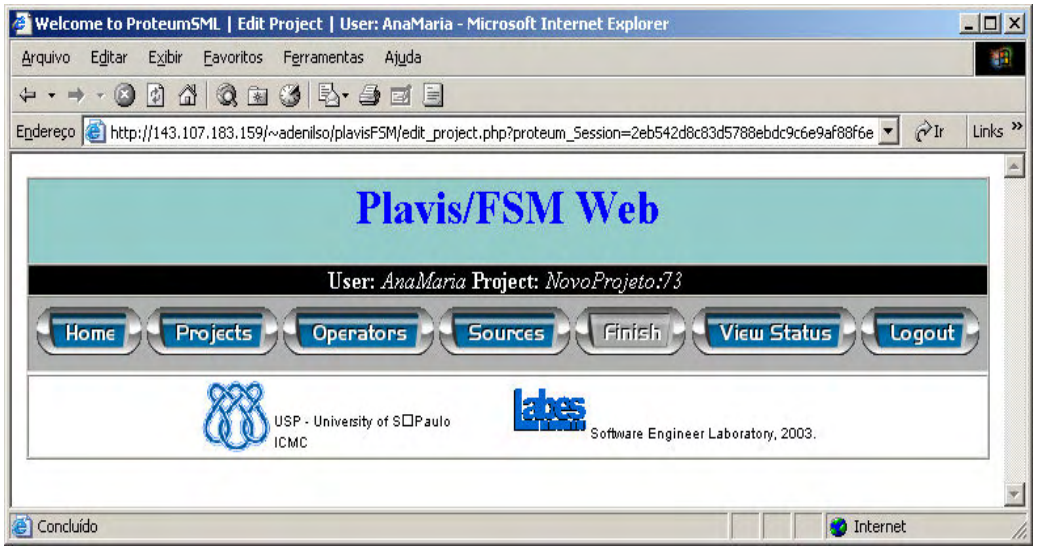

Figura 7 - Tela do Projeto

Botões:

Projects*:* retorna a *Tela Projects.*

Operators*:* abre a *Tela Operators*, possibilitando a escolha dos operadores de mutação que usaremos na geração de mutantes.

Sources*:* exibe a *Tela Sources* para importação ou criação de mefs.

Finish*:* encerra a criação de um projeto, esse botão inicia desabilitado, somente estará habilitado quando os "operadores de mutação" forem escolhidos e o "source" estiver editado, caso contrario permanece desabilitado.

View Status*:* esta tela exibe as informações sobre os mutantes durante a geração dos mesmos, podendo ser usado para acompanhar a geração dos mutantes enquanto finaliza a criação do projeto.

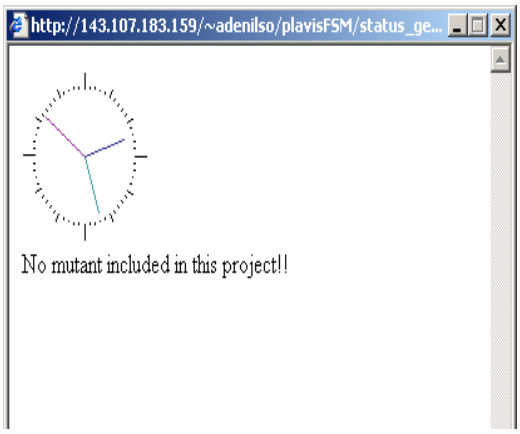

Figura 8 - Tela View Status

5.3 - Tela Operators

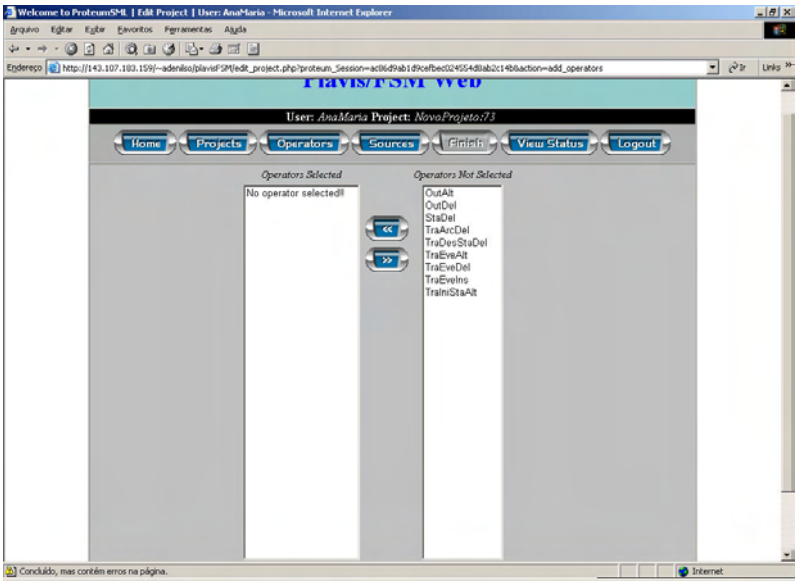

Figura 9 - Tela Operators

Descrição dos operadores:

OutAlt: *saída trocada* - este operador altera a saída gerada pelos eventos, pelas demais saídas existentes. No caso de saídas compostas por vários símbolos, a troca é realizada em cada um deles, desde que a saída que esteja sendo colocada não faça parte das demais saídas já associadas ao mesmo evento.

OutDel: *saída faltando* - este operador remove o símbolo de saída associado a cada evento. No caso do evento possuir vários símbolos de saída associados a ele (saída composta), é retirado um símbolo de cada vez.

StaDel: *estado faltando* - este operador faz uma junção de dois estados em um único, desde que exista um arco conectando os mesmos. O estado resultante dessa junção passa a ser a origem e/ou destino de todos os arcos que tinham como origem e/ou destino os estados que foram agrupados.

TraArcDel: *arco faltando* - este operador exclui um arco da MEF. Considera-se que o arco representa um caminho de um estado para outro; dessa forma, todos os eventos que provocam uma transição entre os estados que estão conectados por esse caminho são excluídos.

TraDesStaDel: *destino trocado* - este operador troca o estado destino associado a cada um dos eventos pelos demais estados existentes na MEF.

TraEveAlt: *evento trocado* - este operador troca cada evento da MEF pelos demais eventos existentes, desde que estes já não provoquem a transição entre os mesmos estados que estão relacionados ao evento que está sendo trocado.

TraEveDel: *evento faltando* - este operador exclui um evento que provoca a transição entre dois estados. No caso desse evento ser o único a provocar essa transição, a alteração gerada por esse operador é a mesma daquela gerado pelo operador arco faltando. No caso da transição entre os estados ser provocada por mais de um evento, os demais eventos continuam existindo.

TraEveIns: *evento extra*- este operador inclui em cada arco da MEF cada um dos outros eventos existentes que não provoca a transição entre os estados relacionados ao arco considerado.

TraIniStaDel: *alteração do estado inicial* - este operador altera o estado inicial da MEF de forma que em cada mutante um dos outros estados passa a ser o estado de inicialização.

Para adicionar um ou mais operador basta seleciona-lo e em seguida clicar no botão de adição da seleção. Para o exemplo iremos adicionar todos os operadores de mutação, assim podemos visualizar a geração de todos os mutantes.

5.4 - Tela Source

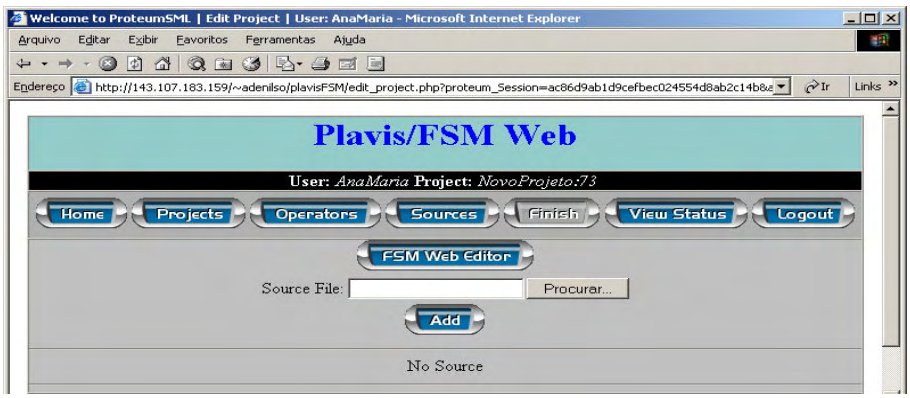

Figura 10 - Tela Sources

Na *Tela Sources* temos a opção de importar uma MEF a partir de um arquivo pronto ou editar uma nova MEF usando o editor. O arquivo importado tem que ter a extensão ".les", para tanto clique no botão *Procurar* e selecione o arquivo. O arquivo adicionado deve aparecer logo abaixo do campo de procura.

Para editarmos uma nova MEF, clicar no botão *FSM Web Editor* e a *Tela FSM Web Editor* será exibida:

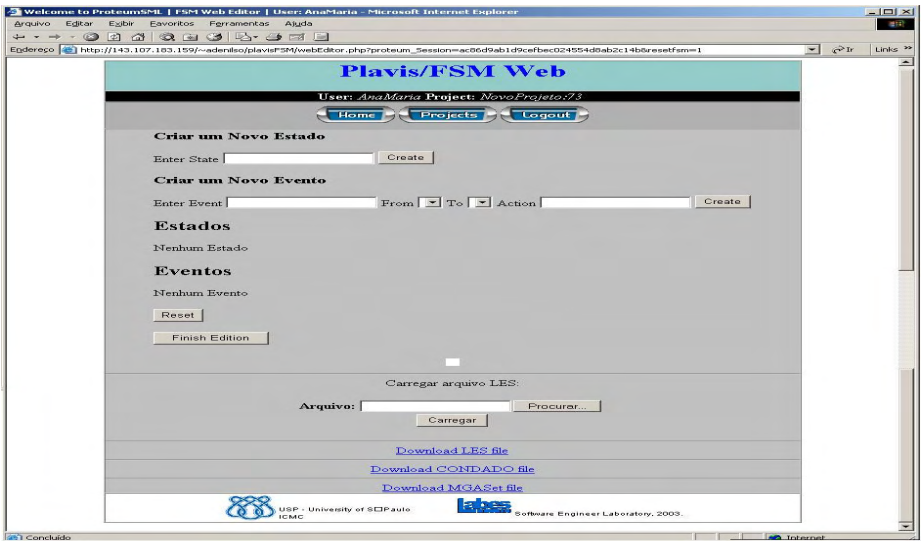

Figura 11 - Tela FSM Web Editor

5.4.1 - Modelo de exemplo

No exemplo usaremos a seguinte especificação:

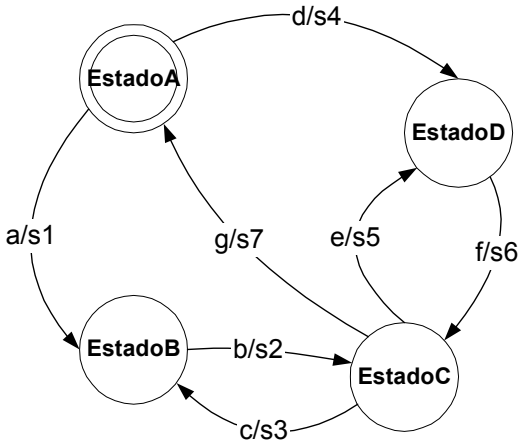

Figura 12 - Especificação Exemplo

Contendo os estados: *EstadoA*, *EstadoB*, *EstadoC*, *EstadoD*, sendo **EstadoA** o estado *inicial e final*. As transições compostas por entrada (evento) e saída (ação) são: *a/s1, b/s2, c/s3, d/s4, e/s5, f/s6, g/s7.*

5.4.2 - Criar estado

Para criação de um novo estado, entre com o nome do estado na caixa de texto *Enter State* e clique em *Create.* Vale lembrar que não é possível usar caracteres especiais.

Exemplo: "Criando o estado que se chama **EstadoA**":

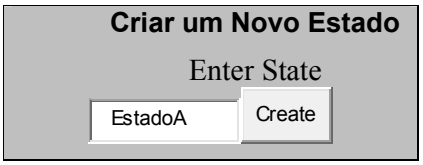

Figura 13 - Criar Estado

Faz se o mesmo para criar outros estados: Estado*B, EstadoC* e *EstadoD.*

5.4.3 - Criar Evento

Para criar um novo evento, entre com o nome do evento na caixa de texto *Enter Event,* na lista *From* escolha o estado de origem, na lista *To* escolha o Estado destino, em *Action* entre com a saída (ação) e clique em *Create.*

Exemplo: "Criando o evento **a** partindo do estado **EstadoA** para o estado **EstadoB**, tendo **s1** como ação":

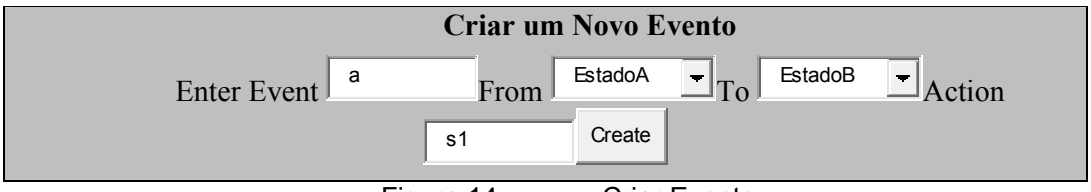

Figura 14 - Criar Evento

Faz se o mesmo para criar outros eventos: *b/s2, c/s3, d/s4, e/s5, f/s6, g/s7*.

5.4.4 - Descrição

Abaixo dos campos de criação de Estados e Eventos, tem uma visualização de todos os estados, eventos e ações.

Estados

Estado A		delete
EstadoB	default	delete
EstadoC	default	delete
EstadD	default	delete

Eventos

evento a	origem EstadoA	destino EstadoB	acao { s1 }	delete
evento b	origem EstadoB	destino EstadoC	acao { s2 }	delete
evento c	origem EstadoC	destino EstadoB	acao { s3 }	delete
evento d	origem EstadoA	destino EstadoD	acao { s4 }	ddelete
evento e	origem EstadoC	destino EstadoD	acao { s5 }	delete
evento f	origem EstadoD	destino EstadoC	acao { s6 }	delete
evento g	origem EstadoC	destino EstadoA	acao { s7 }	delete

Figura 15 - Visualização de Estados e Eventos

Obs.:

1° - no Estado A, o campo *default* não esta disponível pois este é o estado inicial, caso queira mudar o estado inicial, basta clicar em cima do campo *default* de um outro estado, assim no Estado A aparecerá o campo *default* e no estado em que clicou ocultará o campo *default*.

2º - para excluir algum estado ou evento, basta clicar no campo *delete*.

Botões:

Reset: este botão é usado para limpar todos os estados e eventos criados no editor.

Finish Edition: este botão finaliza o *FSM Web Editor*. Somente clique neste botão quando estiver terminado de editar a MEF.

5.4.5 - Imagem gerada

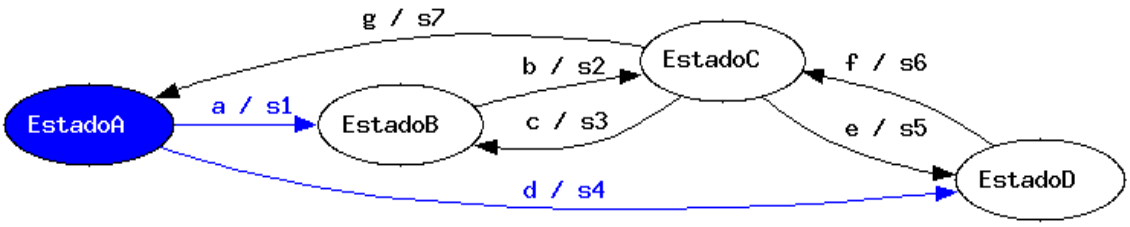

Figura 16 - Imagem Gerada no PLAVIS

A plataforma PLAVIS gera automaticamente a imagem que representa a MEF editada. Caso a imagem não esteja aparecendo, deve se atualizar a página, no *Internet Explorer* basta clicar em *F5* do teclado para que a página seja atualizada e para que a imagem apareça.

5.4.6 - Importar e exportar MEF

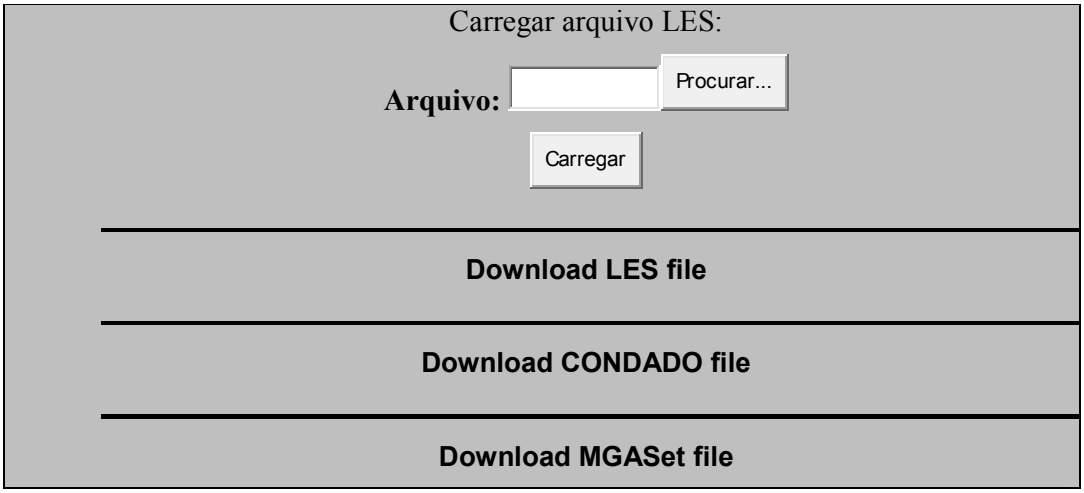

Figura 17 - Exportação e Importação de uma mef

Temos três tipos de opções de exportação dos arquivos MEF:

- .les
- base.pl – *ferramenta CONDADO*
- file.mga – *ferramenta essa MGASet*

Obs.: quando clicamos no botão *Finish Edition* nos terminamos a edição da MEF, não havendo possibilidade de voltar para reedita-la, para tanto a importação é útil caso queiramos reeditar ou usar a mesma MEF em outros projetos.

Para importação de uma MEF, faz se necessário ter um arquivo pronto na extensão *.les* (é aceita apenas a extensão em *.les*). Clique no botão *Procurar...*, escolha o arquivo e clique em *Carregar*. Aparecerão os estados, eventos e a imagem na visualização como descrito anteriormente.

Quando terminado de editar a MEF no *FSM Web Editor*, podemos retornar para *Tela Source.*

5.4.7 - Finalizar a Criação do Projeto

Depois de editar a MEF usando a ferramenta *FSM Web Editor* ou importa-la de um arquivo pronto, a descrição da MEF e visualização estarão disponíveis abaixo do campo de *Procura...,* com a opção de visualização clicando no botão *View*, onde exibirá o código gerado no formato *.les* e a imagem da MEF editada. Também haverá o botão *Remove*, caso deseje remover a MEF.

Exemplo: código gerado na edição da especificação de exemplo:

```
ESTADO FSMmain OU SUBESTADOS EstadoA DEF , EstadoB , EstadoC , EstadoD

ESTADO EstadoA ATOMO

ESTADO EstadoB ATOMO

ESTADO EstadoC ATOMO

ESTADO EstadoD ATOMO

;

EVENTO a ORIGEM EstadoA DESTINO EstadoB ACAO { s1 }

EVENTO b ORIGEM EstadoB DESTINO EstadoC ACAO { s2 }

EVENTO c ORIGEM EstadoC DESTINO EstadoB ACAO { s3 }

EVENTO d ORIGEM EstadoA DESTINO EstadoD ACAO { s4 }

EVENTO e ORIGEM EstadoC DESTINO EstadoD ACAO { s5 }

EVENTO f ORIGEM EstadoD DESTINO EstadoC ACAO { s6 }

EVENTO g ORIGEM EstadoC DESTINO EstadoA ACAO { s7 }

;
```

Figura 18 - Código Gerado Pelo Editor

Com a especificação pronta finalizaremos a edição da MEF clicando em *Finish*. Ao clicar em *Finish* o sistema gera os mutantes conforme os operadores escolhidos. Para acompanhar a geração dos mutantes, clique no botão *View Status*, a *Tela View Status* se abrirá exibindo o número de mutantes gerados.

Obs.: a *Tela View Status* não atualiza automaticamente, é necessário atualizar manualmente a tela do seu browser, no Internet Explorer, basta clicar no botão *F5* do teclado.

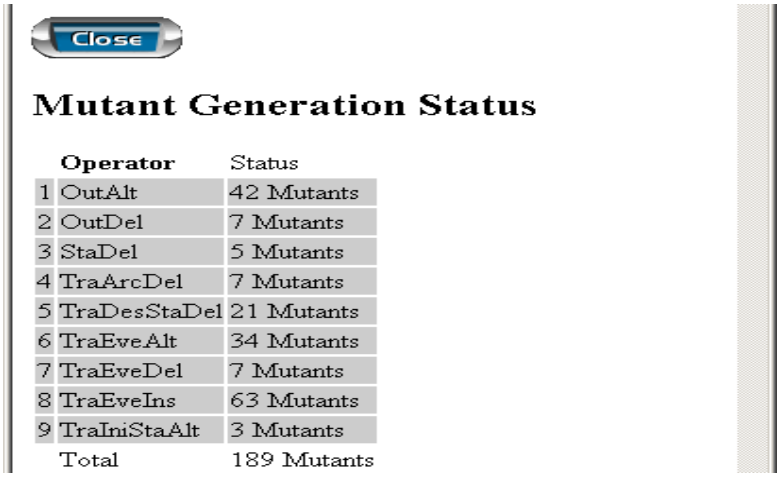

Figura 19 - Tela View Status-Mutantes

A *Tela View Status* mostra o número de mutantes total e o número de mutantes em cada um dos operadores de mutação escolhidos.

6 - Execução dos testes

Voltando a *Tela Projects* os botões *Operators, View Source* e *Finish* se encontram desabilitados, pois a edição do projeto está terminada.

Clicando no botão *Projects*, para ir a *Tela de Sessões*, procure o nome do seu projeto, observe que agora ele esta com ***Status = finished***, isso significa que a edição esta pronta e podemos trabalhar no projeto criado. Clique no nome do projeto e abrirá uma tela diferente da primeira. Veja a *Figura 13. Tela de Sessões*:

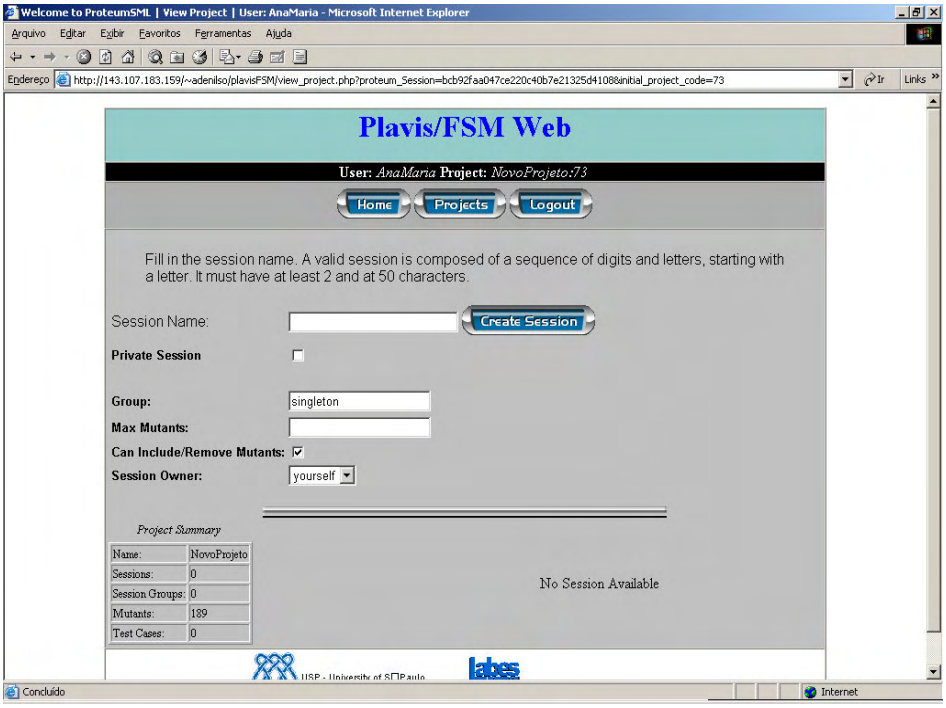

Figura 20 - Tela de Sessões

No canto inferior esquerdo da tela, aparece o *Project Summary*, com algumas informações do projeto atual.

Project Summary	
Name:	NovoProjeto
Sessions:	0
Session Groups:	0
Mutants:	189
Test Cases:	0

Figura 21 - Project Summary

- **Name:** nome do projeto aberto.
- **Sessions**: número de sessões criadas.
- **Session Group**: grupo de sessões criadas.
- **Mutants**: número total de mutantes gerados a partir da MEF.
- **Teste Cases**: número total de casos de teste distintos usados.

6.1 - Criar Sessão

Na *Tela de Sessões*, podemos criar varias sessões para diferentes testes usando a mesma MEF criada anteriormente. Para o exemplo, primeiro criamos uma sessão com o nome **Teste1**. Lembrando que não é possível usar caracteres especiais e nem números como primeiro caractere.

No lugar de *No Session Available* deve aparecer a seguinte lista:

Name	Owner	# Test Cases	# Mutants	Group	Score	
Teste1	AnaMaria			singleton	0.0000	De

Figura 22 - Sessões Criadas

6.1.1 - Descrição das colunas:

Name: nome da sessão criada.

Owner: nome do usuário criador da sessão.

Test Cases: número de *Casos de Testes* gerados ou importados.

Mutants: número de mutantes usados para o teste.

Score: escore coberto pelo teste.

Para acessar as editar e executar o teste basta clicar no nome do mesmo. Exibirá a seguinte tela:

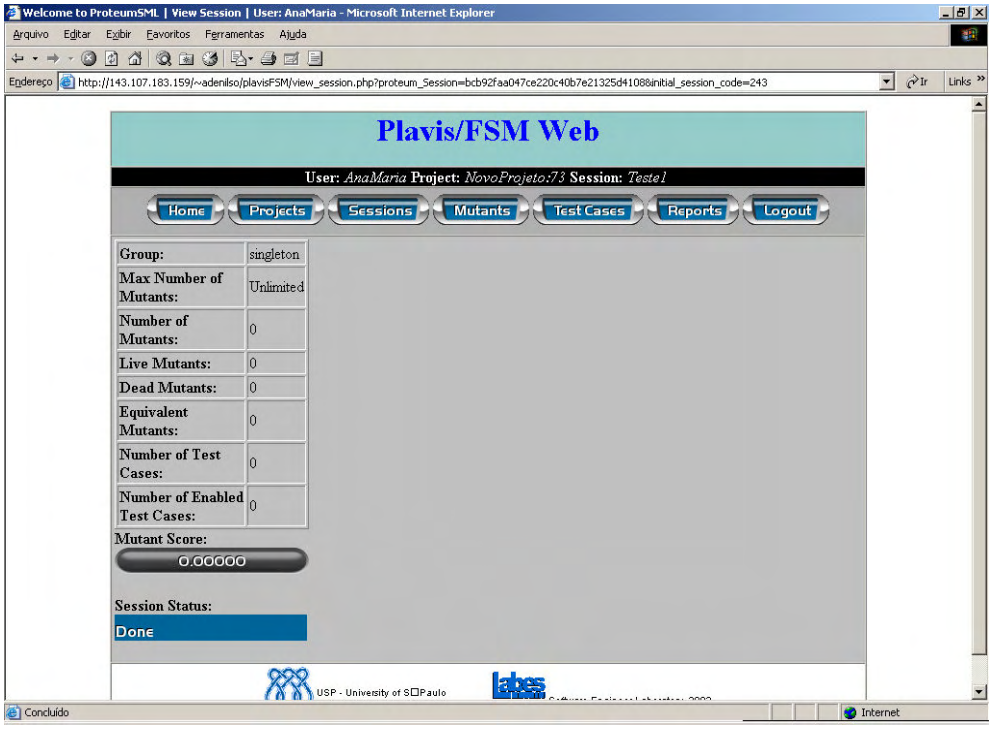

Figura 23 - Tela de teste

No lado esquerdo da tela aparecerão as informações do teste atual.

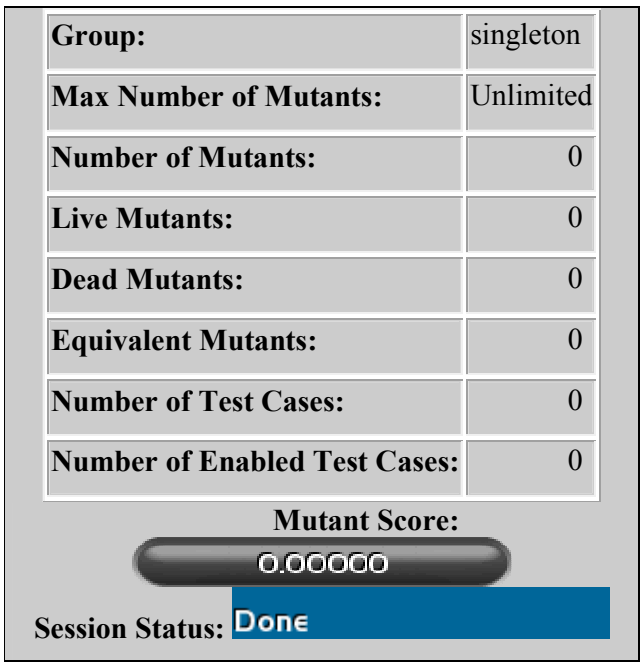

Figura 24 - Informações da Sessão

Max Number of Mutants: número máximo de mutantes a ser usado no teste. O valor padrão é *Unlimited* (Ilimitado), ou seja, todos os mutantes gerados.

Number of Mutantes: número de mutantes usados

Live Mutants: número de mutantes vivos.

Dead Mutants: número de mutantes mortos.

Equivalent Mutants: número de mutantes equivalentes.

Number of Test Cases: número de casos de testes usados.

Number of Enabled Teste Cases: número de casos de testes validos.

Mutant Score: escore coberto pelo teste.

Session Status: processamento da cobertura do teste em %.

Botões:

Sessions: este botão retorna a *Tela de Sessões*.

Mutans: este botão exibe a *Tela de Mutants*, para adicionar os mutantes.

Test Case: este botão exibe a *Tela Casos de Testes*, para geração dos casos de testes usando a ferramenta CONDADO ou o Método W, tendo a possibilidade de importar os casos de teste gerados fora da plataforma PLAVIS.

Mutants:

Ao clicarmos no botão *Mutants* são mostrados os mutantes usados na sessão, quando clicados pela primeira vez não é mostrado nenhum mutante, pois ainda não foram adicionados.

Botões:

Include/Remove Mutants: ao escolhermos o botão *Include/Remove Mutants*, teremos as opções de seleção por *Operator*, seleção por *Source Line* e seleção por *Mutant Number*.

Execute Mutants: após termos gerados ou adicionados casos de testes, pode-se executar os mutantes, para descobrirmos o quanto estes casos de testes adicionados conseguem cobrir dos mutantes gerados.

6.1.2 - Seleção por Operator

Seleção por *Operator*: nesta tela pode visualizar todos os operadores de mutação e a quantidade de mutantes gerados em cada uma deles.

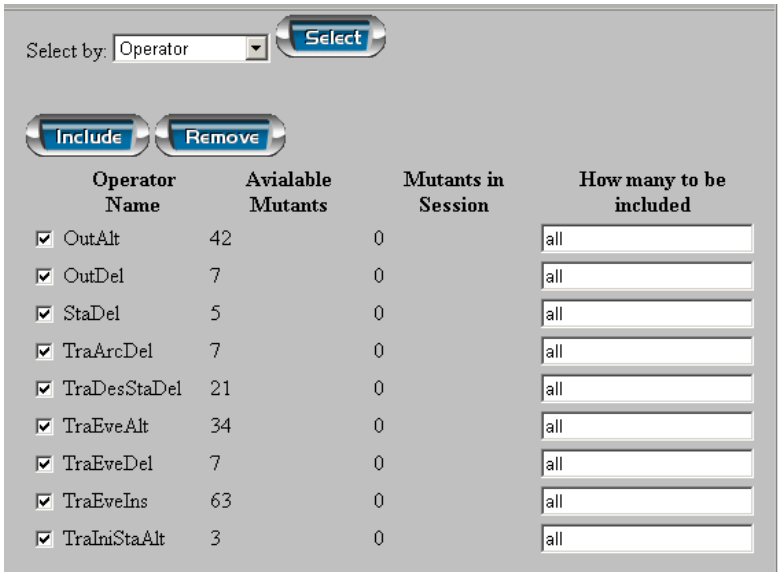

Figura 25 - Tela Seleção por Operator

Descrição das colunas:

Operator Name: nome de cada operador de mutação.

Avialable Mutants: número de mutantes gerados para cada operador de mutação.

Mutants in Session: número de mutantes de cada operador de mutação adicionados a sessão.

How many to be included: quantidade de mutantes de cada operador de mutação que serão adicionados.

Para adicionar o mutante, selecione o operador de mutação desejado e campo *How many to be included*, diga quantos quer adicionar, caso todos mantenha a palavra *all.* Para o exemplo, adicionaremos todos os mutantes gerados.

6.1.3 - Seleção por Source Line

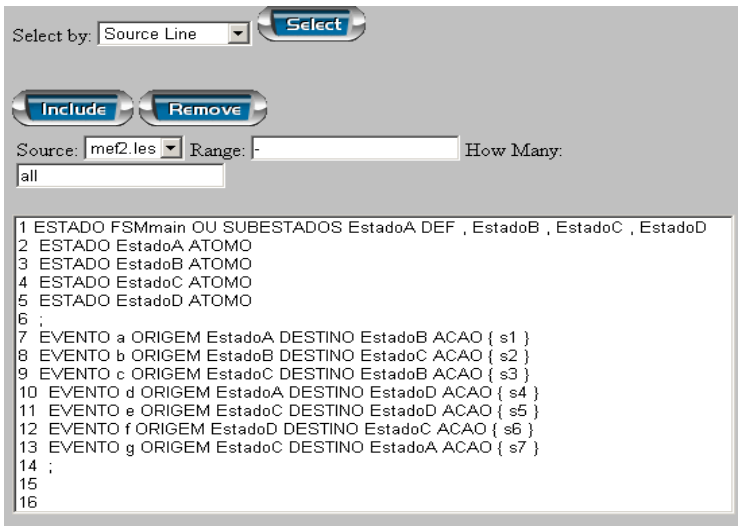

Figura 26 - Tela Seleção por Source Line

Informa-se um conjunto de números das linhas no arquivo LES. Serão considerados apenas os mutantes nos quais pelo menos uma dessas linhas foi alterada. Dessa forma, pode-se, por exemplo, selecionar apenas os mutantes que alteraram um estado específico.

O conjunto de linhas é formado por uma lista de faixas de linhas separadas por vírgula. Uma faixa é formada por um número inicial e um número final. Por exemplo, a especificação1-5,8-20' é formada por duas faixas: uma que vai de 1 a 5 e outra que vai de 8 a 20. Uma faixa de tamanho unitário pode ser representada apenas pelo número. Assim, `7' é equivalente a `7-7'. Tanto o número inicial quanto o número final podem ser omitidos, sendo substituídos por, respectivamente, o número 1 e o número da última linha. Assim, `-10' indica todas as linhas até 10, ao passo que `15-' indica todas as de 15 em diante. Ao se omitir tanto o número inicial quanto o número final, indica-se que todas as linhas devem-se consideradas.

O conteúdo do arquivo LES também é apresentado no formulário. A medida que a faixa é definida, as linhas correspondentes são destacadas. De forma análoga, pode-se selecionar as linhas diretamente (clicando-se sobre ela), e a especificação de linhas será automaticamente calculada.

6.1.4 - Seleção por Mutant Number

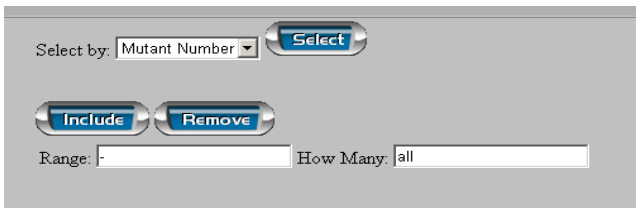

Figura 27 - Tela Seleção por Mutant Number

Cada mutante gerado possui um número que o identifica unicamente entre todos os projetos. Pode-se selecionar um conjunto específico de mutantes, indicando-se um conjunto de números identificadores dos respectivos mutantes. Os números são representados em formato idêntico ao discutido anteriormente para a especificação de linhas (Seção 3.2.3).

6.2 - Test Case

Nesta tela podemos adicionar os casos de testes gerados pela ferramenta CONDADO e pelo Método W, ou ainda, importá-los.

Botões:

Include Test Case: este botão adiciona os casos de testes escolhidos manualmente.

Import W Test Case: este botão gera automaticamente os casos de teste usando o Método W.

Import Condado Test Case: este botão gera automaticamente os casos de teste usando a ferramenta CONDADO.

Import Test Case From File: este botão importa os casos de testes gerados fora da plataforma PLAVIS.

Export Test Case: este botão exporta para o computador um arquivo contendo os casos de testes adicionados à sessão.

Remove All Test: este botão remove todos os casos de testes da sessão.

6.2.1 - Include Test Case

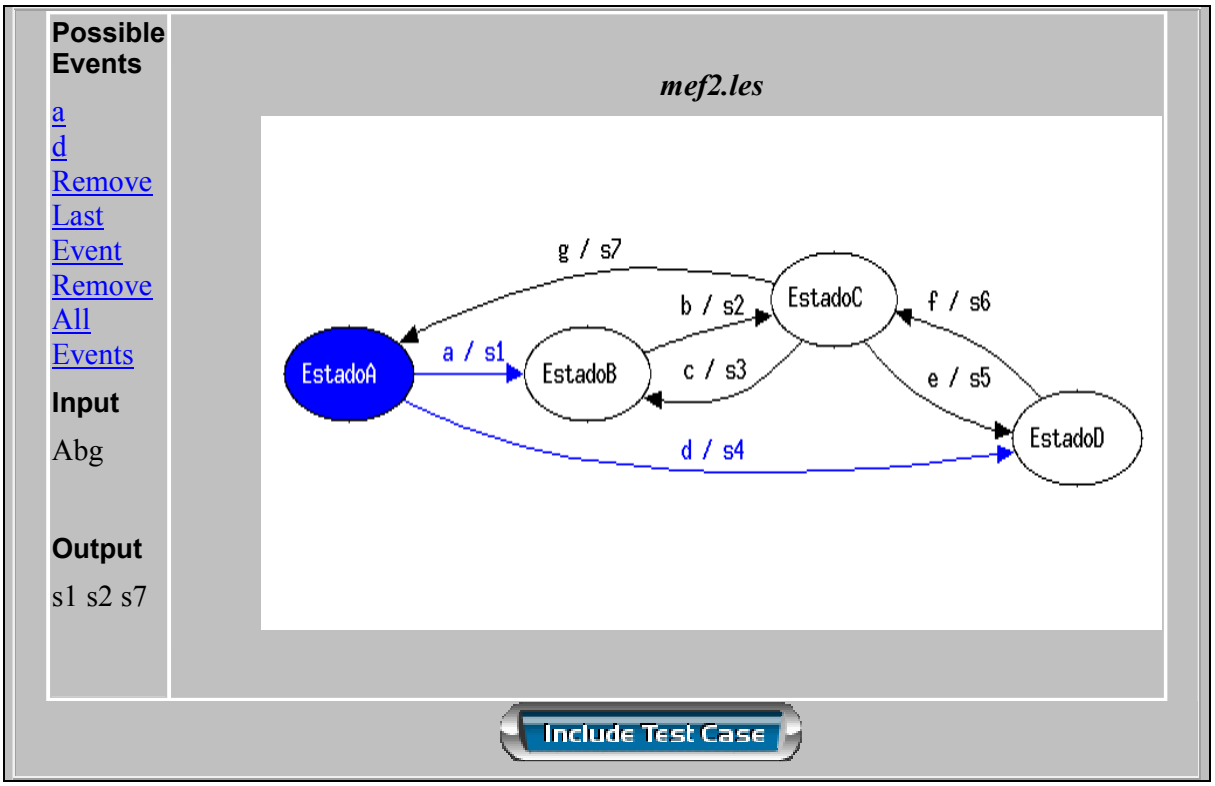

Figura 28 - Tela Include Test Case

Descrição:

Possible Events: nesta coluna são mostrados os possíveis eventos de acontecer, adicionando evento a evento e acompanha-los na figura ao lado.

Inputs: exibe os eventos de entrada adicionados

OutPuts: exibe os eventos de saída adicionados.

Include Test Case: este botão inclui o caso de teste escolhido manualmente.

6.2.2 - Import W Test Case

Este botão *Import W Test Case*, gera automaticamente os casos de testes usando o Método W. Exibindo a quantidade dos casos de testes gerados e na caixa de texto permitindo fornecer quantos destes casos serão usados.

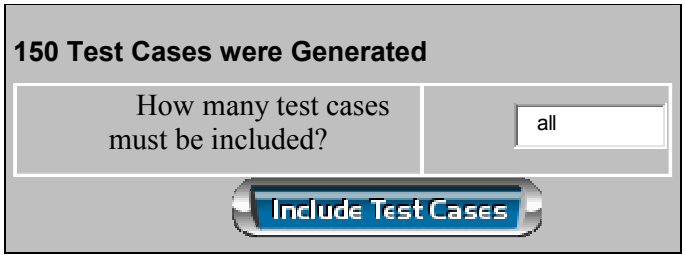

Figura 29 - Mensagem da Geração de Casos de Testes

Clicando no botão *Include Test Cases*, exibirá a mensagem dizendo que incluiu os casos de testes e se gostaria de executar os mutantes, casos já tenha escolhido os

mutantes, basta clicar no botão *Execute Mutants* para começar a execução e no final do processamento lhe mostra qual foi a cobertura dos mutantes com aqueles casos de testes gerados.

Figura 30 - Mensagem de Execução dos Mutantes

Enquanto os mutantes são executados pode-se acompanhar o processo pelo Mutant Score e Session Status.

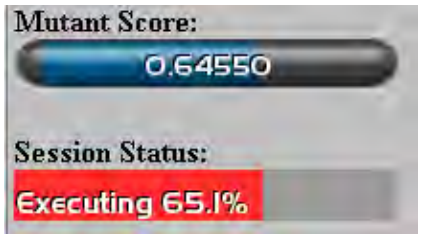

Figura 31 - Status da Execução

Quando terminar as informações do teste a tela deve estar do seguinte modo:

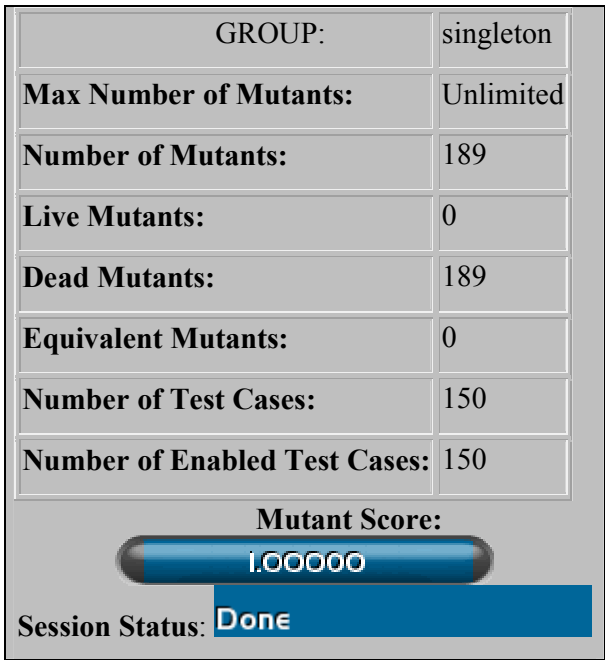

Figura 32 - Informações da Sessão

6.2.3 - Import Condado Test Cases

Para uso deste botão criaremos uma nova sessão, voltando a *Tela de Sessões*, criamos a sessão com o nome **Teste2**.

Obs.: não utilizaremos a sessão *Teste1*, pois esta sessão já esta preenchida com os testes gerados pelo *Método W.*

Com a sessão criada repetiremos os passos *3.2 Mutants* e *3.2.1 Seleção por Operator*. Quando terminado clicamos no botão *Test Cases*, usaremos o recurso *Import Condado Test Cases*.

Essa função gera automaticamente os casos de teste usando a ferramenta CONDADO.

Clicando no botão *Include Test Cases*, confirmará a inclusão dos casos de testes nesta sessão.

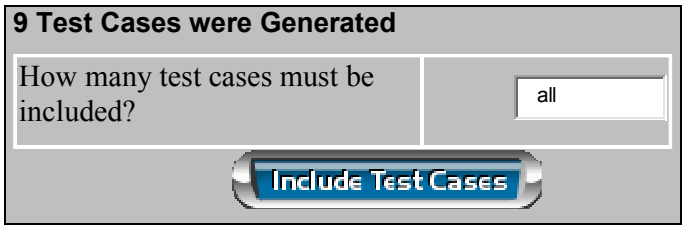

Figura 33 - Mensagem da Geração de Casos de Testes

Como os mutantes já escolhidos, podemos clicar no botão *Execute Mutants*.

Figura 34 - Mensagem de Execução dos Mutantes

Enquanto os mutantes são executados podemos acompanhar o processo pelo *Mutant Score* e *Session Status*.

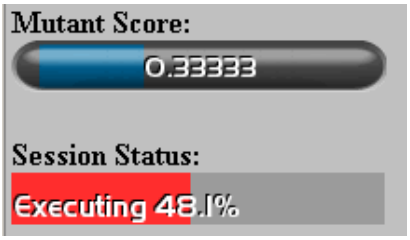

Figura 35 - Status da Execução

Quando terminar as informações do teste a tela deve estar do seguinte modo:

GROUP:	singleton
Max Number of Mutants:	Unlimited
Number of Mutants:	189
Live Mutants:	71
Dead Mutants:	118
Equivalent Mutants:	0
Number of Test Cases:	9

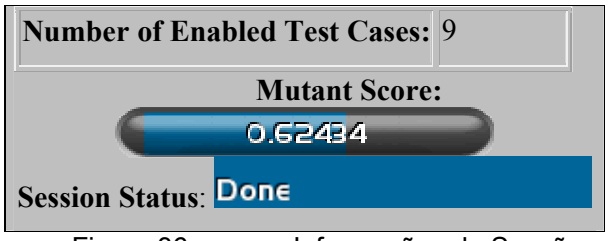

Figura 36 - Informações da Sessão

6.2.4 - Import Test Cases From File

Para uso deste botão criaremos uma nova sessão, voltando para *Tela de Sessões*, daremos o nome de **Teste3** para esta nova sessão.

Com a sessão criada repetiremos os passos *3.2 Mutants* e *3.2.1 Seleção por Operator*. Quando terminado, clicaremos no botão *Test Cases*, usando o botão *Import Test Cases From File*.

Para o exemplo, as seqüências dos casos de testes foram geradas, fora da plataforma PLAVIS, em um arquivo comum de texto do tipo "**.txt**":

Exemplo: com oito casos de teste,

======Inicio do Arquivo======

Seq: a b c b e f g

Seq: a b c b g

Seq: a b e f c b g

Seq: a b e f g

Seq: a b g

Seq: d f c b e f g

Seq: d f c b g

Seq: d f e f c b g

======Fim do Arquivo========

Obs.:

1° Antes de iniciar a seqüência de casos de testes, deve-se colocar a palavra "Seq:", sendo a letra "S" em maiúscula e "eq:" em minúscula;

2° Cada seqüência do caso de teste deve estar na mesma linha separada por espaços em branco;

3° Somente serão usadas as entras na importação, não serão usadas as saídas dos casos de testes.

Clicando no botão *Include Test Cases*, confirmará a inclusão dos casos de teste nesta sessão.

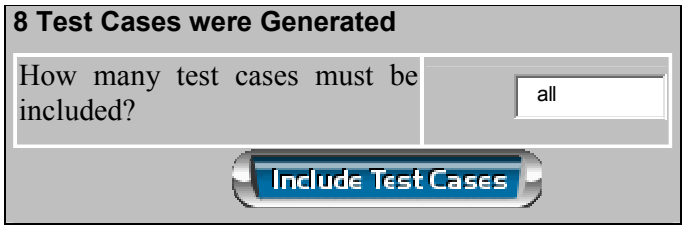

Figura 37 - Mensagem da Geração de Casos de Testes

Com os mutantes previamente escolhidos, clicamos no botão *Execute Mutants*.

Figura 38 - Mensagem de Execução dos Mutantes

Enquanto os mutantes são executados pode-se acompanhar o processo pelo *Mutant Score* e *Session Status*.

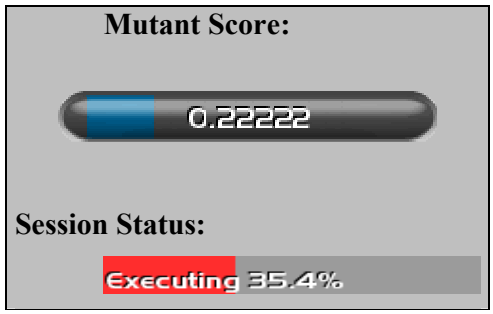

Figura 39 - Status da Execução

Quando terminar as informações do teste a tela deve estar da seguinte maneira:

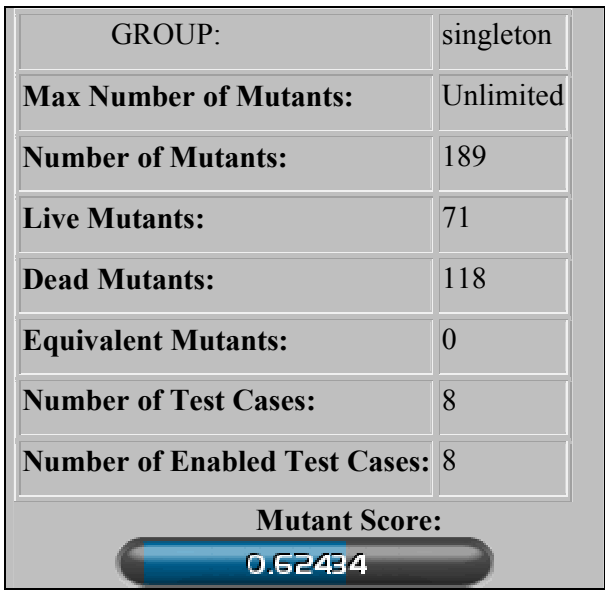

Figura 40 - Informações da Sessão

6.2.5 - Export Test Cases

Este botão exporta para o computador os casos de testes gerados pelas ferramentas CONDADO ou usando o Método W.

O arquivo exportado tem a extensão ".tcs", podendo ser aberto por um editor de texto como por exemplo o *Note Pad* (Bloco de Notas).

6.2.6 - Remove All Test Cases

Este recurso pode ser aplicado para remoção de todos os casos de testes gerados ou adicionados à sessão em andamento.

6.2.7 - Tela de Sessões

Ao terminarmos a *Tela Sessões* estará desta maneira:

Figura 41 - Tela Sessões

Para criar um novo projeto devem-se repetir os passos a partir do passo *2.2 Criação de um projeto*.

Agradecimentos:

Ao coordenador do projeto: José Carlos Maldonado. Aos pesquisadores que forneceram suas ferramentas: Eliane Martins e Ana Ambrosio, Sandra Fabri, Ricardo Anido, Adenilso Simão. A todos os participantes do projeto PLAVIS que de uma forma e outra contribuíram para a realização desta Plataforma. Aos órgãos financiadores: Capes e CNPq.